第４章　数と文字とストリーム入出力

4-1　数と文字

数と文字や基数等に関しては、「情報処理基礎Ｉ」で詳しく学習したはずですから、ここでは詳細は省略します。

■ エスケープシーケンス（拡張表記）

□ エスケープシーケンス**¥a** を使うと、アラーム（ベル）を鳴らすことができます。

□ エスケープシーケンス**¥n** は、改行文字でしたね。

□ 一重引用符

'A'や'x'などとは異なり、一重引用符文字'は'''と表すことはできません。エスケープシーケンス¥'を用いて、'¥''と表記します。

■ 引用符'を横に連続表示

```
/*
    for文＋文字による描画　'を横に連続表示
*/

#include  <iostream.h>

int main(void)
{
    int  i;
    int  no;                    // 表示する個数

    cout << "いくつ表示しますか：";
    cin >> no;

    for (i = 0; i < no; i++)
        cout << '¥'';
    cout << '¥n';

    return (0);
}
```

```
いくつ表示しますか：15
'''''''''''''''
```

□ 二重引用符

同様に、文字列リテラルの中で二重引用符文字を使いたい場合は、エスケープシーケンス¥"を用います。"ABC"と表示するプログラムを以下に示します。

■ "ABC"と表示

```
/*
    "ABC"と表示
*/

#include  <iostream.h>

int main(void)
{
    cout << "¥"ABC¥"";

    return (0);
}
```

```
"ABC"
```

□ 復帰

エスケープシーケンス**¥r** によって、カーソルは、その行の先頭に位置づけられます。したがって、そこから書き始めることによって、既に何かが書かれている行を『書きかえる』ことができます（次ページのプログラム）。

※ プログラムは、難しいので全部を理解する必要はありません。
※ 関数 sleep は、x ミリ秒だけ、時間をつぶす働きをします。
※ 表示速度の調整は、以下の 100 の値を書きかえましょう。

```
sleep(100);
```

※ また、実行は Cntrl + C によって中断します。

文字列を電光掲示板のように、流しながら表示するプログラムを以下に示します。

■ 文字列をテロップ表示

```
/*
    文字列を流しながら表示する
*/

#include  <time.h>
#include  <string.h>
#include  <iostream.h>

/*--- xミリ秒経過するのを待つ ---*/
int sleep(unsigned long x)
{
    clock_t  s = clock();
    clock_t  c;
    do {
        if ((c = clock()) == (clock_t)-1)      /* エラー */
            return (0);
    } while (1000UL * (c - s) / CLOCKS_PER_SEC <= x);
    return (1);
}

int main(void)
{
    int  cnt = 0;
    char mes[] = "BohYoh Shibata ";
    int  len = strlen(mes);

    while (1) {
        cout << '¥r';     // カーソルを行の先頭へ

        for (int i = 0; i < len; i++) {
            if (cnt + i < len)
                cout << mes[cnt + i] << flush;
            else
                cout << mes[cnt + i - len] << flush;
        }

        sleep(100);           // この値を書き換えると速度調整できる

        if (cnt < len - 1)
            cnt++;
        else
            cnt = 0;
    }
    return (0);
}
```

```
BohYoh Shibata
      ↓
ohYoh Shibata B
      ↓
hYoh Shibata Bo
      ↓
Yoh Shibata Boh
      ↓
oh Shibata BohY
      ↓
h Shibata BohYo
      ↓
 Shibata BohYoh
      ↓
Shibata BohYoh
      ↓
hibata BohYoh S
      ↓
ibata BohYoh Sh
      ↓
      :
```

□ 後退

エスケープシーケンス¥b によって、カーソルを一つ戻すことができます。123456789 を順に表示した後に、逆から順に消すプログラムを示します。

■123456789 を順に表示した後に、逆から順に消す

```
/*
    123456789を順に表示した後に逆から消す
*/

#include  <time.h>
#include  <string.h>
#include  <iostream.h>

/*--- xミリ秒経過するのを待つ ---*/
int sleep(unsigned long x)
{
    clock_t  s = clock();
    clock_t  c;
    do {
        if ((c = clock()) == (clock_t)-1)      /* エラー */
            return (0);
    } while (1000UL * (c - s) / CLOCKS_PER_SEC <= x);
    return (1);
}

int main(void)
{
    int i;

    for (i = 1; i <= 9; i++) {
        char x = '0' + i;
        cout << x << flush;
        sleep(1000);              // 1秒停止
    }

    for (i = 1; i <= 9; i++) {
        cout << "¥b ¥b" << flush;
        sleep(1000);              // 1秒停止
    }

    return (0);
}
```

```
↓
1
↓
12
↓
123
↓
1234
↓
12345
↓
123456
↓
1234567
↓
12345678
↓
123456789
↓
12345678
↓
1234567
↓
123456
↓
12345
↓
1234
↓
123
↓
12
↓
1
```

4-2　ストリームへの出力における書式化

■ フィールド幅指定

表示の最低幅を n 桁に指定する。

```
cout.width(n)
```

表示を必要な桁数に指定する（3 桁の数値は 3 桁で、4 桁の数値は 4 桁で)。

```
cout.width(0)
```

■ フィル文字指定

表示がフィールド幅に満たない際に埋める文字を'x'に指定する。

```
cout.fill('x')
```

■ 精度指定

浮動小数点出力における精度を n に指定する。

```
cout.precision(n)
```

■ 書式状態の指定

各種の書式状態を設定する。

```
cout.setf(flag)     cout.setf(flag, field)
```

各種の書式状態を解除する。

```
cout.unsetf(flag)   cout.unsetf(flag, field)
```

【例】

```
cout.setf(ios::oct, ios::basefield)          // 8 進
cout.setf(ios::dec, ios::basefield)          // 10 進
cout.setf(ios::hex, ios::basefield)          // 16 進

cout.setf(ios::shobase)                      // 基数の区別
cout.setf(ios::showpoint)                    // 浮動小数点の出力
cout.setf(ios::showpos)                      // 正符号の表示

cout.setf(ios::scientific, ios::floatfield) // 指数形式

cout.setf(ios::left, ios::adjustfield)       // 左詰め
cout.setf(ios::right, ios::adjustfield)      // 右詰め
cout.setf(ios::internal, ios::adjustfield)  // 符号と数値の間にフィル文字
```

4-3　処理子

■ 基数の設定

値 x を 8 進数／10 進数／16 進数で表示する。

```
cout << oct << x;          // ８進数
cout << dec << x;          // 10 進数
cout << hex << x;          // 16 進数
```

■ 改行の挿入

改行を挿入し、さらにバッファをフラッシュします（文字を 1 文字ずつ出力するとスピードが低下するので、出力すべき文字はバッファに蓄えられており、何らかのタイミングで掃き出すことによってスピードアップを図るのが一般的です。バッファに残っている文字を全て掃き出します)。

```
cout << endl;
```

■ フィールド幅の指定

表示の最低幅を n 桁に指定する。

```
cout << setw(n)
```

表示を必要な桁数に指定する（3 桁の数値は 3 桁で、4 桁の数値は 4 桁で)。

```
cout << setw(0)
```

■ 精度指定

```
cout << setprecision(n)
```

■ 書式状態の指定

```
cout << setiosflag(flag)
cout << resetiosflag(flag)
```

九九の表を演算しながら表示しましょう。
※いったん計算式を表示した後に、¥b を利用して式を消します。

■ 九九の表を演算しながら表示

```
/*
    九九算（掛け算の途中を表示）
*/

#include  <time.h>
#include  <iomanip.h>
#include  <iostream.h>

/*--- xミリ秒経過するのを待つ ---*/
int sleep(unsigned long x)
{
    clock_t  s = clock();
    clock_t  c;
    do {
        if ((c = clock()) == (clock_t)-1)
            return (0);
    } while (1000UL * (c - s) / CLOCKS_PER_SEC <= x);
    return (1);
}

int main(void)
{
    int  i, j;

    for (i = 1; i <= 9; i++) {
        for (j = 1; j <= 9; j++) {
            cout << setw(3) << i << "□*" << setw(3) << j;
            sleep(2000);
            cout << "¥b¥b¥b¥b¥b¥b¥b";
            cout << setw(3) << i * j;
        }
        cout << "□□□□□¥n";
    }
    return (0);
}
```

```
 1 * 1
     ↓
 1
     ↓
 1  1 * 2
     ↓
 1  2
     ↓
 1  2  1 * 3
     ↓
 1  2  3
     ↓
 1  2  3  1 * 4
     ↓
 1  2  3  4
     ↓
 1  2  3  4  1 * 5
     ↓
 1  2  3  4  5
     ↓
（中略）
     ↓
 1  2  3  4  5  6  7  8 9
 2 * 1
     ↓
 1  2  3  4  5  6  7  8 9
 2
     ↓
```

注意：□はスペース文字です。

■ 演習問題

◆ 以下に示すのは、100 から 200 までの整数を 8 進数、10 進数、16 進数で表示するプログラムの部分である。なお、表示はそれぞれ 5 桁の幅内に行うものとする。

```
for (int i = (1); i <= (2); (3)) {
    cout << (4) << (5)(5) << i
         << (6) << (5)(5) << i
         << (7) << (5)(5) << i << '¥n';
}
```

```
 144   100   64
 145   101   65
 146   102   66
        :
```

◆ 以下のプログラムは、0 から 255 までの 16 進数を、5 桁の幅で各行に 8 個ずつ表示するプログラムである。

```
#include<iostream.h>

int main(void)
{
    for (int i = (8); i < (9); (10)) {
        cout << (11) << (12) << i;
        if ( (13) )
            cout << '¥n';
    }
    return (0);
}
```

(1)	
(2)	
(3)	
(4)	
(5)	
(6)	
(7)	
(8)	
(9)	
(10)	
(11)	
(12)	
(13)	

■ プログラム作成演習

(4-1) 右に示すように、1 から 9 までの整数をずらしながら表示するプログラムを作成せよ。

※二重ループを使うことなく実現せよ。

```
1
  2
    3
（以下省略）
```

(4-2) 右に示すように、1 から 9 までの整数をずらしながら（その前に+を）表示するプログラムを作成せよ。

```
1
+2
++3
（以下省略）
```

(4-3) 右に示すように、段数を読み込み、123 をずらしながら表示するプログラムを作成せよ。

```
何段ですか：4↵
123
 123
  123
   123
```

(4-4) 下に示すように、読み込んだ整数値の符号を反転した値を表示するプログラムを作成せよ。なお、値の正負に関わらず、+あるいは-の符号を表示すること。

setiosflag 処理子を使うこと。すなわち setiosflag(ios::showpos)を挿入すること。

※ x の符号を反転した値は、$-x$ である。

```
整数値を入力せよ：4↵
符号を反転した値は-4です。
```

```
整数値を入力せよ：-20↵
符号を反転した値は+20です。
```